Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 100 (2016) 1235 – 1241

Procedia Computer Science

Conference on ENTERprise Information Systems / International Conference on Project MANagement / Conference on Health and Social Care Information Systems and Technologies, CENTERIS / ProjMAN / HCist 2016, October 5-7, 2016

Project Scope Partitioning by Clustering Features into Releases of Long R&D Projects

Ran Etgar[a]*, Roy Gelbard[b], Yuval Cohen[c]

[a]*Open University, Raanana, Israel*
[b]*Bar-Ilan University, Ramat-Gan 5900, Israel*
[c]*Afeka College for Engineering, Tel-Aviv 69107, Israel*

Abstract

R&D projects are characterized by a long planning horizon, which entails the policy of release management. The intermediate releases enable the organization to maximize the value for a given investment. Myopic version of this problem is known as the Next Release Problem (NRP). A central issue addressed by these projects is determining which features should be included in the next release. The choice of features impacts the value of the release, but also impacts the required workload, and future development of other features. NRP can be expanded to include the later releases. This problem is NP-hard and thus cannot be solved analytically. In this work we apply a simple clustering algorithm, based on novel similarity coefficients to reduce complexity. Our goal is to provide a near-optimal yet simple method for quantitatively determining the feature content of all project releases.

Peer-review under responsibility of the organizing committee of CENTERIS 2016

Keywords: Scope of work; Scheuling; version; releases

1. Introduction

The current paper tackles the problem of planning the scope of research and development (R&D) project releases over time. To achieve higher flexibility and to better satisfy actual customer requirements, there is an increasing

* Corresponding author. Tel.: +972-52-8695155; fax: +972-8-9702714.
E-mail address: ran.etgar@gmail.com

Peer-review under responsibility of the organizing committee of CENTERIS 2016
doi:10.1016/j.procs.2016.09.154

tendency to run the development project in an incremental fashion [1]. The research deals with determining the content (scope of work – SOW) of each release in the planning horizon over time by clustering product features. Strategic release planning (sometimes referred to as road-mapping) is an important phase of the requirements engineering process performed at product level [2]. Release planning is a complex problem, as appropriate understanding of planning objectives and technical constraints are required for a good release plan [3]. Typically, the firm is able to pre-commit to a schedule of technology releases [4]. Despite the wide need of strategic release planning, developed methodology includes mostly qualitative tools and lacks quantitative capability for planning and scheduling [5].

Clustering techniques were used in the field of software engineering to solve important software engineering problems in the context of reflexion analysis, software evolution, and information recovery [6]. This research provides a novel tool to cluster the various features to the planned releases, by simultaneously considering technological constraints, resource availability and market value [7]. Thus, enabling the stakeholders to make decisions based on quantitative facts rather than subjective less-exact methods.

2. Problem Description

Determining requirements for upcoming releases is a complex process. The typical case is that the marketing department wishes to include as many features as possible in the early releases [8], however there are more requirements than can be implemented since the capacity of available resources is limited [9]. Furthermore, the features are not independent entities and the scheduler has to deal with technological precedence. Therefore, there is a need to consider the value gained by including a feature in the upcoming release (or the one after, or the one after that…) against the value of other features that compete on the same resources. The scheduler work is to decide which features to include in which version release.

Obviously, features included in the first release provide higher value than including the same features in later releases (due to time value of money, competitive market, pressure from users etc.). To reflect this discounting phenomenon, there is a need to assign corresponding value to the version releases. These values set a descending series (for example, the nearest release is set with value 1. The next release has value 0.9, indicating that a feature included in the second release provide only 90% of the value it could have provided had it been scheduled to the first release. The third release is assigned value of 0.85, thus indicating an even lower value, etc.). The features are not independent, but have precedence constraints, as depicted in Figure 1. Thus, the problem becomes more complex. Feature 12 (for example) may have high value and require low resources, thus making it lucrative to be included in early release, but the precedence network dictates that it to be included only after the development of features 3,6,7,8,9 and 11.

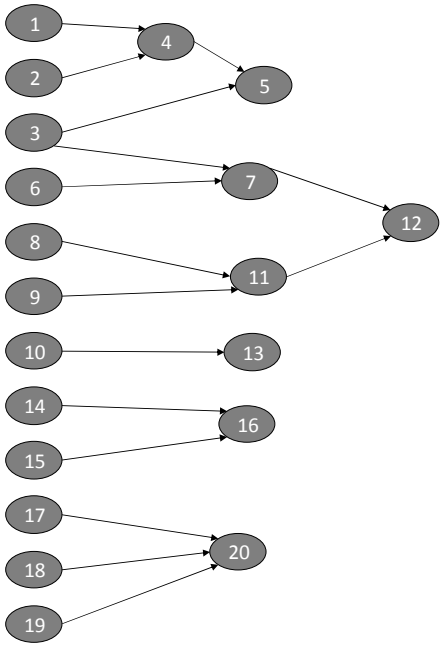

Figure 1 - Project network. Each node indicates a feature

It is simple to show that this problem in NP-hard (by reduction to either bin packing problem or to Resource Constrained Project Scheduling Problem - RCPSP), but a good solution may be achieved by using hierarchical cluster analysis techniques.

3. Clustering Features

While being NP-hard, the problem of assigning features to releases can be solved quite simply if we resort to clustering techniques. The complexity of the problem is reduced significantly by clustering the features to "release size" clusters.

Fayyad et al. [10] defined four main building blocks in a clustering task: Attribute† selection, the Clustering algorithm, Validation of the results, and Interpretation of the results. By applying these blocks, clustering allow us to decide which features should be assigned with others to the same release.

The first step is to use the following similarity (and dissimilarity) aspects of the features to build the attributes:

- Precedence based similarity – The most obvious similarity coefficient is based on the technological precedence constraints. A feature cannot be added to a cluster (representing release) before its preceding ones are scheduled.
- Resource based dissimilarity – When building the clusters, it is important to understand that the features compete with each other on the same resource pool. Therefore, a negative similarity coefficient can be derived from the resource requirements – the more two feature share the same resource demands, the less

† Fayyad used the term "feature". For obvious reason the term "attribute" is used instead.

they can be clustered together - $\rho_{i,j} = \frac{r_i + r_j}{R}$ (where r_i is the resource requirement and R is the resource availability).

- Predecessor based similarity – another aspect of the network, is the sharing of the same predecessors - $\pi_{i,j} = \frac{pred_{i,j}}{pred_i + pred_j + pred_{i,j}}$ (where $pred_j$ is the number of predecessors of j and $pred_{i,j}$ is the number of common predecessors of both i and j)
- Successor based similarity – the final aspect of the network, is sharing the same successors – $\eta_{i,j} = \frac{Succ_{i,j}}{Succ_i + Succ_j + Succ_{i,j}}$, similar to $\pi_{i,j}$

From these 4 similarity components we developed coefficients for feature clustering. By clustering the features together, the complexity of the problem is reduced from $O(n!)$, where n is the number of features, to $O(m^2)$, where m is the number of releases. The first of these coefficient is a go/no go (i.e. compulsory) and the rest are calculated by trying different weights.

Typically, for the various clusters formed, and their implementation in the original environment is not defined [11]. However, for the specific purpose of this algorithm, there is a clear cutting point for the cluster size – the resource constraints for the specific release.

4. Example

Consider the network depicted in Figure 1. Assume there are 3 types of resources involved (e.g. programmers, QA personnel, system analysts). Finally, let us also consider the resource demands that are exhibited in Table 1. In this network 3 types of resource are used. For simplicity sake it is assumed that each feature requires only one type of resource.

Table 1: Resource demand

Feature	Resource	Level	Value	Feature	Resource	Level	Value
1	1	2	3	11	2	2	2
2	3	2	3	12	3	2	5
3	2	1	2	13	2	2	3
4	2	1	1	14	2	1	2
5	1	2	4	15	2	1	1
6	1	2	3	16	3	1	5
7	1	1	2	17	1	2	3
8	2	1	5	18	2	3	2
9	3	2	1	19	3	3	3
10	2	2	1	20	2	3	2

While applying the clustering method, we assumed for simplicity, that all resources have the same constant availability level - (6 units).

Of course we allocate discount values for each release as discussed above. In this example we set the second release to be 10% less than the first, and the third to be further 10% less (i.e. values: 1, 0.9, 0.8).

The next stage is to calculate the similarity coefficient between the features. These coefficients are depicted in Table 2. Based on the similarity coefficient and UPGMA hierarchical clustering, the dendrogram depicted in Figure 2 can be calculated.

As explained, the resource constraints set an external cutting point for the dendrogram. Applying the resource constraints on the dendrogram, results in 'cutting' the left part (containing 1,2,4,3,5,8,14,15,16) from the middle (containing 18,19,20) and the right (containing 6,7,9,11,12,10,13,17) resulting in the releases depicted in Figure 3.

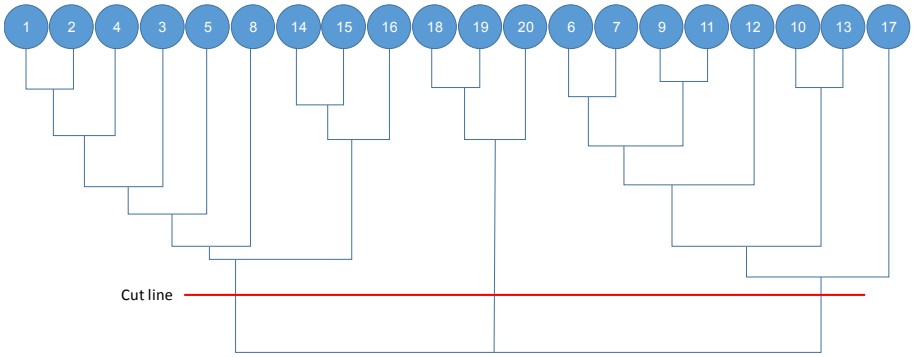

Figure 2 - Features Dendrogram

Table 2 - Similarity Matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1		0.67	0.50	0.83	0.17	0.17	0.25	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.17	0.50	0.50	0.50
2	0.67		0.50	0.83	0.50	0.50	0.50	0.50	0.17	0.50	0.50	0.17	0.50	0.50	0.50	0.25	0.50	0.50	0.08	0.50
3	0.50	0.50		0.46	0.83	0.63	0.79	0.33	0.50	0.25	0.25	0.50	0.25	0.33	0.33	0.50	0.50	0.17	0.50	0.17
4	0.83	0.83	0.46		0.85	0.50	0.50	0.33	0.50	0.25	0.25	0.50	0.25	0.33	0.33	0.50	0.50	0.17	0.50	0.17
5	0.17	0.50	0.83	0.85		0.17	0.35	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.17	0.50	0.50	0.50
6	0.17	0.50	0.63	0.50	0.17		0.54	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.17	0.50	0.50	0.50
7	0.25	0.50	0.79	0.50	0.35	0.54		0.50	0.50	0.50	0.63	0.85	0.50	0.50	0.50	0.50	0.25	0.50	0.50	0.50
8	0.50	0.50	0.33	0.33	0.50	0.50	0.50		0.67	0.25	0.58	0.50	0.25	0.33	0.33	0.50	0.50	0.17	0.50	0.17
9	0.50	0.17	0.50	0.50	0.50	0.50	0.50	0.67		0.50	0.83	0.17	0.50	0.50	0.50	0.25	0.50	0.50	0.08	0.50
10	0.50	0.50	0.25	0.25	0.50	0.50	0.50	0.25	0.50		0.17	0.50	0.67	0.25	0.25	0.50	0.50	0.08	0.50	0.08
11	0.50	0.50	0.25	0.25	0.50	0.50	0.63	0.58	0.83	0.17		0.85	0.17	0.25	0.25	0.50	0.50	0.08	0.50	0.08
12	0.50	0.17	0.50	0.50	0.50	0.50	0.85	0.50	0.17	0.50	0.85		0.50	0.50	0.50	0.25	0.50	0.50	0.08	0.50
13	0.50	0.50	0.25	0.25	0.50	0.50	0.50	0.25	0.50	0.67	0.17	0.50		0.25	0.25	0.50	0.50	0.08	0.50	0.08
14	0.50	0.50	0.33	0.33	0.50	0.50	0.50	0.33	0.50	0.25	0.25	0.50	0.25		0.50	0.92	0.50	0.17	0.50	0.17
15	0.50	0.50	0.33	0.33	0.50	0.50	0.50	0.33	0.50	0.25	0.25	0.50	0.25	0.50		0.67	0.50	0.17	0.50	0.17
16	0.50	0.25	0.50	0.50	0.50	0.50	0.50	0.50	0.25	0.50	0.50	0.25	0.50	0.92	0.67		0.50	0.50	0.17	0.50
17	0.17	0.50	0.50	0.50	0.17	0.17	0.25	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50		0.67	0.67	0.63
18	0.50	0.50	0.17	0.17	0.50	0.50	0.50	0.17	0.50	0.08	0.08	0.50	0.08	0.17	0.17	0.50	0.67		0.67	0.38
19	0.50	0.08	0.50	0.50	0.50	0.50	0.50	0.50	0.08	0.50	0.50	0.08	0.50	0.50	0.50	0.17	0.67	0.67		0.88
20	0.50	0.50	0.17	0.17	0.50	0.50	0.50	0.17	0.50	0.08	0.08	0.50	0.08	0.17	0.17	0.50	0.63	0.38	0.88	

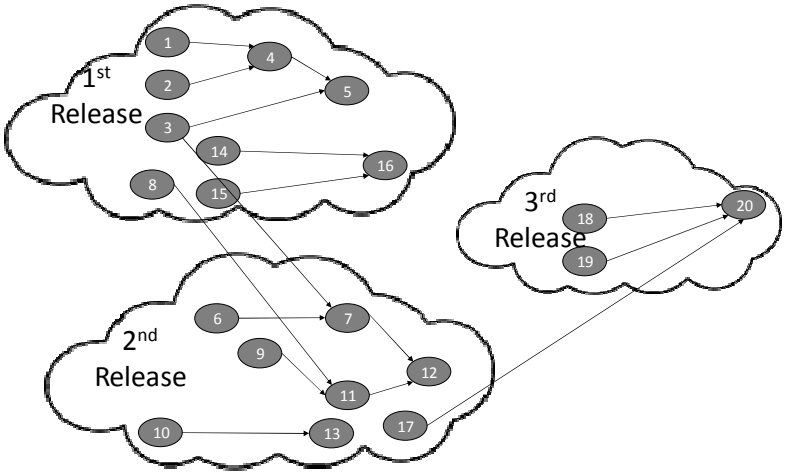

Figure 3 - Feature allocation to releases (with precedence relations)

In Figure 3, the three releases initiated by the cut line of Figure 2 is depicted. The 1st release corresponds to the left part of the dendrogram. The second is the right part and the 3rd is the middle part.

5. Discussion and Conclusion

The proposed algorithm can provide a simple and easy tool for

- Increasing management ability to solve the next release problem. Rather than use solely intuitive tool, management can rely also on quantitative tools.
- Explore several development alternatives: by manipulating the variables (mainly, the value of the different features), project managers can examine different development strategies, and analyze the potential outcomes.
- Improve communication between different stakeholders in the project. As each stakeholder is in charge of a different part of the overall process, it is simple to do what-if analyses and save valuable management time.
- The combination of all the above advantages should yield R&D projects that better suit the organization targets and adhere to their strategic work plans.

It remains for future research to compare this method of clustering features to releases to other heuristic methods.

References

[1] D. Greer and G. Ruhe, "Software release planning: an evolutionary and iterative approach," *Information and Software Technology*, vol. 46, p. 243–253, 2004.

[2] M. Svahnberg, T. Gorschek, R. Feldt, R. Torkar, S. Bin Saleem and M. U. Shafique, "A systematic review on strategic release planning models," *Information and Software Technology*, vol. 52, pp. 237-247, 2010.

[3] G. Ruhe and M. O. Saliu, "The art and science of software release planning," *IEEE Software*, vol. 6, no. 22, pp. 47-53, 2005.

[4] I. Lobel, J. Patel, G. Vulcano and J. Zhang, "Optimizing Product Launches in the Presence of Strategic Consumers," *Management Science*, p. Published On line, 2015.

[5] Y. Geum, S. Lee and Y. Park, "Combining technology roadmap and system dynamics simulation to support scenario-planning: A case of car-sharing service," *Computers & Industrial Engineering*, no. 71, pp. 37-49, 2014.

[6] M. Shtern and V. Tzerpos, "ClusteringMethodologies for Software Engineering," *Advances in Software Engineering*, vol. 2012, pp. 1-18,

2012.

[7] M. Harman, J. Krinke, I. Medina-Bulo, F. Palomo-Lozano, J. Ren and S. Yoo, "Exact Scalable Sensitivity Analysis for the Next Release Problem," *ACM Transactions on Software Engineering and Methodology,* vol. 23, no. 2, pp. 19:1-19:31, 2014.

[8] M. van den Akker, S. Brinkkemper, G. Diepen and J. Versendaal, "Software product release planning through optimization and what-if analysis," *Information and Software Technology,* vol. 50, p. 101–111, 2008.

[9] J. J. Durillo, Y. Zhang, E. Alba, M. Harman and A. J. Nebro, "A study of the bi-objective next release problem," *Empirical Software Engineering,* vol. 16, no. 1, pp. 29-60, 2011.

[10] U. Fayyad, G. Piatetsky-Shapiro and P. Smyth, "From Data Mining to Knowledge Discovery in Databases," *AI Magazine,* vol. 17, no. 3, pp. 37-54, 1996.

[11] R. Gelbard, O. Goldman and I. Spiegler, "Investigating diversity of clustering methods: An empirical comparison," *Data & Knowledge Engineering,* no. 63, p. 155–166, 2007.